Lecture I — From Autocomplete to Agents

AI-Assisted Programming for PhD Researchers

Dr. Tobias Vlćek
Helmut Schmidt University — September 2026

Welcome

About This Course

- Three days, hands-on from the first hour
- You build a real data pipeline, then your own project
- Accredited through a final presentation, not an exam
- The goal: use AI to code *well*, not just fast

About Me

- **Field:** Optimizing and simulating complex systems
- **Languages:** Julia, Python, and Rust
- **Interests:** Modelling, Simulations, Machine Learning
- **Teaching:** OR, Algorithms, and Programming
- **More:** tobiasvlcek.com

The Three Days

- **Day 1** — Foundations and the core agent loop
- **Day 2** — Discipline: verification, safety, extension
- **Day 3** — Your own project, then presentations

What You Will Deliver

A final presentation covering four things:

- A **demo** of what you built
- A **reflection** on your process
- The **challenges** you hit
- The **insights** you take away

This Website

- Everything you need lives here — no separate handouts
- Slides open as a deck via the RevealJS button
- Labs are step-by-step pages you follow at your pace
- A course chatbot sits in the sidebar for questions

Who Are You?

Quick intro round.

> **Note**
>
> Tell us three things:
>
> - Your **name**
> - Your **field**
> - One piece of **code you fight with**

Why This Course

Research Runs on Code

- Most PhDs write research software with little to no formal software training [1]
- Hannay's survey of nearly 2,000 scientists showed learning was mostly informal, from peers
- That was already hard — *before* AI entered the loop

What Changed

- **2021**: autocomplete suggests the next line as you type
- **2023**: chat answers questions in a side window
- **2025+**: agents read your files and run commands
- A capability jump, and a widening discipline gap

The Promise and the Trap

- **Promise:** hours saved on boilerplate, debugging, and learning
- **Trap:** code you don't understand ends up in your thesis

. . .

- A gap widens between those who use AI well and those who lean on it as a crutch [2]

The Tool Landscape

Three Kinds of AI Coding Tools

- **Autocomplete**: inline, at the keystroke level
- **Chat**: you copy and paste, it answers
- **Agents**: they act inside your repository

Autocomplete

- Completes code as you type
- Zed edit predictions are what you got with the student plan
- Great for boilerplate
- Useless for design decisions

Chat

- Paste your code, get an answer back
- No repository access — *you* are the clipboard

- Still useful for concepts and quick explanations

Agents
- Read files, edit them, run commands, iterate toward a goal
- Live in your terminal or your editor
- This is the course's focus: agentic coding

The 2026 Market
- **IDE-integrated:** Copilot (sign-up freeze noted), Cursor, Windsurf
- **Vendor CLIs:** Claude Code, Codex, Gemini CLI
- **Open source:** OpenCode, Aider, Goose

Orientation only — no detail needed today.

Our Stack, and Why
- **Zed**: fast editor, free student plan
- **OpenCode**: open source, works with any model
- **Mistral**: free tier, EU-hosted, no training on opted-out data

Learn the workflow, not the tool. These concepts transfer to every other agent.

How Agents Work

A Language Model in a Loop
- The model **proposes** an action
- A **tool runs** it: read, edit, or shell command
- The **result feeds back** into the model
- Repeat until the goal is met

Nothing magical, and no memory between sessions.

Context Is Everything
- The model only knows what is in its **context window**
- That means: your prompt, files it read, command output
- Nothing else — not your screen, not your intent
- Garbage in, garbage out

Where It Breaks
- **Hallucination**: plausible is not the same as true
- **Stale knowledge**: training has a cutoff date
- **Long-context degradation**: quality drops in huge sessions

The answer is verification — we build that on Day 2.

The Evidence

Faster — Sometimes, Somewhere
- Three RCTs, n=4,867: **+26% completed tasks** [3]

- Juniors gained the most
- **+55.8%** on a greenfield HTTP-server task [4]

These are real gains — on the right kind of work.

...But Slower Where It Counts

- Experienced devs on their *own* mature codebases:

. . .

- **19% slower** — while believing they were about 20% *faster* [5]
- The perceived speedup and the real one point opposite ways

Does AI Assistance Hurt Learning?

- Anthropic ran a randomized controlled trial on coding skill

. . .

- AI group scored **50%** on a mastery quiz vs **67%** hand-coding [6]
- The largest gap was on **debugging** questions

Comprehension Debt

- Code that works today but nobody understands tomorrow [7]
- Like technical debt, but for *understanding*
- The interest comes due during revisions and review
- You pay it when a reviewer asks "why does this work?"

It Depends How You Use It

- Same study: conceptual questions and explain-back **preserved** learning; passive "just fix it" did not
- At scale: unguarded AI cut exam scores, a guardrailed tutor did not (high-school maths RCT) [8]

This course *is* those guardrails.

AI Code Has More Bugs Than You Think

- About **45%** of AI-generated samples carry OWASP-class vulnerabilities [9]
- AI-authored PRs: **~1.7× more issues overall**, up to **2.74×** for XSS specifically (correlational) [10]

Day 2 is built around catching exactly this.

Our Working Rules

Rule 1 — Explain Before You Accept

- The era's shift is from *writing* code to *reading and evaluating* it [11]
- You must be able to explain every change the agent made
- Labs have "Explain it" boxes for exactly this [12]

Rule 2 — Verify, Then Trust

- Never believe the word "done"
- **Run** it
- **Test** it
- **Read** the diff

Rule 3 — You Own Every Line

- "The AI wrote it" is not a defense — not in your thesis, your paper, or this course
- **Task stewardship:** delegate the labor, keep the accountability [13]

Rule 4 — Small Steps, Frequent Commits

- Small, verified increments beat one giant diff
- Every step stays reviewable and reversible
- Git makes this cheap — that's our next session

Continue Your Journey

Next Up

- **Version Control with Git** — 10:45
- Lecture II — Version Control with Git
- Course literature and references

Bibliography

[1] J. E. Hannay, C. MacLeod, J. Singer, H. P. Langtangen, D. Pfahl, and G. Wilson, "How do scientists develop and use scientific software?," in *2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, May 2009, pp. 1–8. doi: 10.1109/SECSE.2009.5069155.

[2] J. Prather *et al.*, "The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers," in *Proceedings of the 2024 ACM Conference on International Computing Education Research - Volume 1*, Melbourne VIC Australia: ACM, Aug. 2024, pp. 469–486. doi: 10.1145/3632620.3671116.

[3] K. Z. Cui, M. Demirer, S. Jaffe, L. Musolff, S. Peng, and T. Salz, "The Effects of Generative AI on High-Skilled Work: Evidence from Three Field Experiments with Software Developers," *Management Science*, p. mnsc.2025.00535, Feb. 2026, doi: 10.1287/mnsc.2025.00535.

[4] S. Peng, E. Kalliamvakou, P. Cihon, and M. Demirer, "The Impact of AI on Developer Productivity: Evidence from GitHub Copilot." Accessed: July 09, 2026. [Online]. Available: https://arxiv.org/abs/2302.06590

[5] J. Becker, N. Rush, E. Barnes, and D. Rein, "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity." Accessed: July 09, 2026. [Online]. Available: https://arxiv.org/abs/2507.09089

[6] J. H. Shen and A. Tamkin, “How AI Impacts Skill Formation.” Accessed: July 09, 2026. [Online]. Available: https://arxiv.org/abs/2601.20245

[7] A. Osmani, “Comprehension Debt: The Hidden Cost of AI-Generated Code.” Accessed: July 09, 2026. [Online]. Available: https://addyosmani.com/blog/comprehension-debt/

[8] H. Bastani, O. Bastani, A. Sungu, H. Ge, Ö. Kabakcı, and R. Mariman, “Generative AI without guardrails can harm learning: Evidence from high school mathematics,” *Proceedings of the National Academy of Sciences*, vol. 122, no. 26, p. e2422633122, July 2025, doi: 10.1073/pnas.2422633122.

[9] Veracode, “2025 GenAI Code Security Report.” Accessed: July 09, 2026. [Online]. Available: https://www.veracode.com/wp-content/uploads/2025_GenAI_Code_Security_Report_Final.pdf

[10] “AI vs human code gen report: AI code creates 1.7x more issues.” Accessed: July 09, 2026. [Online]. Available: https://coderabbit.ai/blog/state-of-ai-vs-human-code-generation-report

[11] P. Denny *et al.*, “Computing Education in the Era of Generative AI,” *Communications of the ACM*, vol. 67, no. 2, pp. 56–67, Feb. 2024, doi: 10.1145/3624720.

[12] D. H. Smith and C. Zilles, “Code Generation Based Grading: Evaluating an Auto-grading Mechanism for "Explain-in-Plain-English" Questions,” in *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*, Milan Italy: ACM, July 2024, pp. 171–177. doi: 10.1145/3649217.3653582.

[13] H.-P. (. Lee *et al.*, “The Impact of Generative AI on Critical Thinking: Self-Reported Reductions in Cognitive Effort and Confidence Effects From a Survey of Knowledge Workers,” in *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, Yokohama Japan: ACM, Apr. 2025, pp. 1–22. doi: 10.1145/3706598.3713778.